```
import pandas as pd
from pandas.plotting import scatter_matrix
import numpy as np
import matplotlib.pyplot as plt
import os
from imblearn.over_sampling import ADASYN
from collections import Counter
import seaborn as sn
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn import metrics
```

```
pd.set_option('display.max_row',None)
pd.set_option('display.max_column',None)
```

```
def plot_confusion_matrix(cm, classes, title, cmap):
    "function for plotting confusion matrix"
    plt.clf()
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    classnames = classes
    plt.title(title)
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    tick_marks = np.arange(len(classnames))
    plt.xticks(tick_marks, classnames, rotation=45)
    plt.yticks(tick_marks, classnames)
    s = [['TN','FP'], ['FN', 'TP']]
    for i in range(2):
        for j in range(2):
            plt.text(j, i, str(s[i][j])+" = "+str(cm[i][j]))
    plt.show()

def plot_roc_auc(model_list, X_test, y_test):
    "a function to plot roc_auc"
    fig, ax = plt.subplots(figsize=(8, 6))
    for model_name, model in model_list:
        y_score = model.predict_proba(X_test)[:, 1]
        fpr, tpr, _ = metrics.roc_curve(y_test, y_score)
        roc_auc = metrics.auc(fpr, tpr)
        plt.plot(fpr, tpr, lw=2, label=model_name + ' (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')
    plt.legend(loc="lower right")
    plt.show()
```

```
import seaborn as sns

# Set up plotting parameters
%matplotlib inline
custom_style = "dark"
custom_palette = "colorblind"

# Set seaborn style and palette
sns.set_style(custom_style)
sns.set_palette(custom_palette)
```

```
df = pd.read_csv("creditcard.csv")
```

```
df.head(5)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V1
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.9913
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.4890
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.7172

```
# for determining the number of records in the dataset
print('The dataset contains {0} rows and {1} columns.'.format(df.shape[0], df.shape[1]))
```

```
The dataset contains 23858 rows and 31 columns.
```

```
# check for missing values and data types of the columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23858 entries, 0 to 23857
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Time    23858 non-null  int64  
 1   V1      23858 non-null  float64
 2   V2      23858 non-null  float64
 3   V3      23858 non-null  float64
 4   V4      23858 non-null  float64
 5   V5      23858 non-null  float64
 6   V6      23858 non-null  float64
 7   V7      23858 non-null  float64
 8   V8      23858 non-null  float64
 9   V9      23858 non-null  float64
 10  V10     23858 non-null  float64
 11  V11     23858 non-null  float64
 12  V12     23858 non-null  float64
 13  V13     23858 non-null  float64
 14  V14     23858 non-null  float64
 15  V15     23858 non-null  float64
 16  V16     23858 non-null  float64
 17  V17     23858 non-null  float64
 18  V18     23858 non-null  float64
 19  V19     23858 non-null  float64
 20  V20     23858 non-null  float64
 21  V21     23858 non-null  float64
 22  V22     23857 non-null  float64
 23  V23     23857 non-null  float64
 24  V24     23857 non-null  float64
 25  V25     23857 non-null  float64
 26  V26     23857 non-null  float64
 27  V27     23857 non-null  float64
 28  V28     23857 non-null  float64
 29  Amount  23857 non-null  float64
 30  Class   23857 non-null  float64
dtypes: float64(30), int64(1)
memory usage: 5.6 MB
```

```
print('count of Normal transactions: ', df['Class'].value_counts().values[0])
print('count of Fraudulent transactions: ', df['Class'].value_counts().values[1])
```

```
count of Normal transactions:  23769
count of Fraudulent transactions:  88
```

```
# feature data (predictors)
features = df.iloc[:, :-1]
# label class
labels = df['Class']
```

```
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

```
X_train, X_test, y_train, y_test = train_test_split(scaled_features,labels, test_size=0.30, random_state=42)
```

```
adasyn_sampler = ADASYN(random_state=42)
print('Original dataset shape {}'.format(Counter(y_train)))
X_resampled, y_resampled = adasyn_sampler.fit_resample(X_train, y_train)
print('Resampled dataset shape {}'.format(Counter(y_resampled)))
```

```
Original dataset shape Counter({0.0: 16636, 1.0: 64})
Resampled dataset shape Counter({0.0: 16636, 1.0: 16635})
```

```
X_train_resampled, y_train_resampled = X_resampled, y_resampled

# Train LogisticRegression Model
logistic_regression_classifier = LogisticRegression()
logistic_regression_classifier.fit(X_train_resampled, y_train_resampled)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

```
# Train Decision Tree Model
random_forest_classifier = RandomForestClassifier(random_state=0)
random_forest_classifier.fit(X_train, y_train)
```

```
▾       RandomForestClassifier
RandomForestClassifier(random_state=0)
```

```
# Train Bernoulli Naive Baye Model
bernoulli_nb_classifier = BernoulliNB()
bernoulli_nb_classifier.fit(X_train, y_train)
```

```
▾ BernoulliNB
BernoulliNB()
```

```
modlist = [('RandomForest Classifier', random_forest_classifier), ('LogisticRegression', logistic_regression_classifier),
           ('Naive Baye Classifier', bernoulli_nb_classifier)]

models = [j for j in modlist]

print()
print('========================= Model Evaluation Results ========================' "\n")

for model_name, model in models:
    scores = cross_val_score(model, X_train, y_train, cv=10)
    accuracy = metrics.accuracy_score(y_train, model.predict(X_train))
    confusion_matrix = metrics.confusion_matrix(y_train, model.predict(X_train))
    classification = metrics.classification_report(y_train, model.predict(X_train))
    print(f'===== {model_name} =====')
    print()
    print("Cross Validation Mean Score: ", '{}%'.format(np.round(scores.mean(), 3) * 100))
    print()
    print("Model Accuracy: ", '{}%'.format(np.round(accuracy, 3) * 100))
    print()
    print("Confusion Matrix:" "\n", confusion_matrix)
    print()
    print("Classification Report:" "\n", classification)
    print()
```

```
========================= Model Evaluation Results ========================

===== RandomForest Classifier =====

Cross Validation Mean Score:  99.9%

Model Accuracy:  100.0%

Confusion Matrix:
 [[16636     0]
 [    0    64]]
```

```
Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     16636
         1.0       1.00      1.00      1.00        64

    accuracy                           1.00     16700
   macro avg       1.00      1.00      1.00     16700
weighted avg       1.00      1.00      1.00     16700


/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1)
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1)
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1)
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1)
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1)
```

```
df.describe()
```

	Time	V1	V2	V3	V4	V5
count	23858.000000	23858.000000	23858.000000	23858.000000	23858.000000	23858.000000
mean	18213.370609	-0.239141	0.198892	0.727022	0.248619	-0.188428
std	11377.032190	1.894219	1.533073	1.724887	1.440938	1.439894
min	0.000000	-30.552380	-40.978852	-31.103685	-5.172595	-42.147898
25%	6624.750000	-0.959528	-0.376134	0.287941	-0.658457	-0.767634
50%	20564.000000	-0.288644	0.192491	0.874426	0.216440	-0.218348
75%	29010.250000	1.164867	0.843146	1.505467	1.122367	0.325281
max	32954.000000	1.960497	16.713389	4.101716	11.927512	34.099309